INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS
CENTRO DE PREVISÃO DE TEMPO E ESTUDOS CLIMÁTICOS

Introdução ao Fortran

Luis Thiago Lucci

Cachoeira Paulista, SP

Introdução

- IBM Mathematical FORmula TRANslation System
- Elaborado especificamente para aplicações científicas
- Grande número de bibliotecas numéricas construído.
- Compiladores (especialmente os livres) são fáceis de encontrar em diferentes ambientes.

Introdução

- Compiladores

  – Programa que a partir de um código fonte escrito em uma linguagem compilada, cria um programa semanticamente equivalentepara outra linguagem:código objeto.

  Compiladores mais utilizados:

  - gfortran(gnu)
  - ifort (Intel Fortran)
  - pgf90 (pgi – Portland Group)

Introdução

- O Fortran77 está obsoleto em relação às linguagens atuais e aos recursos existentes

  Formato fixo:

  Linhas da posição 7 a 72;

  Variáveis até 6 caracteres.

- O Fortran 90possui novos recursos na definição de um vetor, matriz alocação de memória dinâmica, apontadores, etc

  Formato livre:

  132 caracteres por linha;

  Variáveis até 31 caracteres.

- Os compiladores disponíveis distinguem entre esses dois formatos através da extensão do nome do arquivo .f ou .F para form fixo e .f90 ou .F90 formato livre.

Conceitos básicos

- Caracteres válidos
  - Caracteres alfabéticos:
    sem distinção de entre maiúscula e minúscula
  - Caracteres numéricos: todos os dígitos de 0 até 9
  - Caracteres especiais

    <branco> ` <apóstrofo>

    . <ponto> “ <aspas>

    , <vírgula> $: & ?

    + - * / = ( )
- Comentários !

Conceitos básicos

- A estrutura básica de um programa FORTRAN pode ser descrita como:

```
PROGRAM  <nome do programa>
< declarações    >
< comandos       >
END PROGRAM  <nome do programa>
```

Declarações - conjunto de comandos que definem os dados a serem usados.

Comandos - conjunto de comandos que serão executados, na ordem em que aparecem listados e atendendo a quaisquer desvios em seu fluxo de execução.

Variáveis

- Uma variável FORTRAN é um nome para uma localização de memória.
  - Devem iniciar por uma letra
  - Podem conter letras e números
  - Não faz distinção entre letras maiúsculas e minúsculas

    VAR = var = Var
  - Tamanho máximo de uma variável no fortran 90 é 31 caracteres

Declaração de variáveis

- Implicitamente
  - Variáveis iniciadas com I, J, K, L, M, ou N - variáveis inteiras
  - Variáveis iniciadas com outras letras - variáveis reais
- Explicitamente
  - INTEGER - varáveis inteiras
  - REAL - variáveis reais
  - DOUBLE PRECISION - varáveis de dupla precisão
  - COMPLEX - variáveis complexas
  - LOGICAL - variáveis lógicas
  - CHARACTER - variáveis caracteres

* Recomenda-se definir todas as variáveis

implicit none - obriga a declaração de variáveis no bloco

Declaração de variáveis

- Variáveis do tipo INTEGER

  Este tipo de variável armazena apenas a parte inteira de um número, exemplos de números inteiros válidos, também denominados literais são:

  123, 89312, 5

  As declarações básicas de variáveis de tipo inteiro são:

  Fortran 77: INTEGER <lista de variáveis>

  exemplo: INTEGER contador01

  Fortran 90/95: INTEGER :: <lista de variáveis>

  exemplo: INTEGER :: contador01

Declaração de variáveis

- Variáveis do tipo REAL

  O tipo de variável real é composto de quatro partes, assim dispostas: uma parte inteira (com ou sem sinal), um ponto decimal, uma parte fracionária e /ou um expoente, também com ou sem sinal.. Exemplos de literais reais são:

  -10.6E-11 (representando $-10{,}6 \times 10^{-11}$)

  1E-1 (representando 10^{-1} ou 0,1)

  3.141592653, 1., -0.1

  As declarações básicas de variáveis de tipo inteiro são:

  Fortran 77: REAL <lista de variáveis>

  exemplo: REAL Pi

  Fortran 90/95: REAL :: <lista de variáveis>

  exemplo: REAL :: Pi

Declaração de variáveis

- Variáveis do tipo COMPLEX

Um elemento complexo é um valor numérico com parte real e parte imaginária. Qualquer das partes pode ser um inteiro ou um real. Uma constante complexa é representada através de dois elementos entre parêntesis e separados por virgula, representando o primeiro a parte real e o segundo a parte imaginária, (real, imaginária).

(1. , 3.2) (representando 1 - 3,2i)

(1. , 0.99E-2) (representando $1 - 0{,}99 \times 10^{-2}i$)

As declarações básicas de variáveis de tipo inteiro são:

Fortran 77: COMPLEX <lista de variáveis>

exemplo: COMPLEX A

Fortran 90/95: COMPLEX :: <lista de variáveis>

exemplo: COMPLEX :: A

Declaração de variáveis

- Variáveis do tipo CHARACTER

O tipo padrão consiste em um conjunto de caracteres contidos em um par de apóstrofes ou aspas. Os apóstrofes (‘) ou aspas (“) servem c delimitadores dos literais de caractere e não são considerados parte integrante do conjunto. Ao contrário das normas usuais, um espaço em branco é diferente de dois ou mais. Exemplos de literais de caractere:

‘bom Dia’, ‘bomDia’, “BRASIL”, “Fortran 90”

As declarações básicas de variáveis de tipo inteiro são:

Fortran 77: CHARACTER (<comprimento>) <lista de variáveis>

exemplo: CHARACTER (30) Nome

Fortran 90/95: CHARACTER (len= <comprimento>) :: <lista de variávei

exemplo: CHARACTER (len=30) :: Nome

Declaração de variáveis

- Variáveis do tipo LOGICAL

  O tipo lógico define variáveis lógicas. Uma variável lógica só pode assumir dois valores, verdadeiro e falso. A representação dos dois estados possíveis de uma variável lógica são:

  .TRUE. ⇒ Verdadeiro

  .FALSE. ⇒ Falso

  As declarações básicas de variáveis de tipo inteiro são:

  Fortran 77: LOGICAL <lista de variáveis>

  exemplo: LOGICAL chave01

  Fortran 90/95: LOGICAL :: <lista de variáveis>

  exemplo: LOGICAL :: chave01

Declaração de variáveis

- Declaração de uma matriz ou vetor (array)

DIMENSION ➡ Específica o número de dimensões e o número de elementos em cada dimensão da variável.

Exemplos:

INTEGER, DIMENSION(10) :: A ➡ 1 dimensão e 10 elementos

REAL, DIMENSION(2,4) :: B ➡ 2 dimensões uma com 2 elementos (linhas) e outra com 4 elementos (co

Declaração de variáveis

- Declaração de um parâmetro

  PARAMETER ➡ variável não poderá ter seu valor alterado

  Exemplos:

  INTEGER, PARAMETER :: M=10, N=20, NZ=M*N-M

  REAL, PARAMETER :: PI=3.1415926

Declaração de variáveis

- Alocação de Matrizes

  O Fortran90 permite a alocação dinâmica de memória. Para isso será necessário utilizar os comandos ALLOCATABLE, ALLOCATE e DEALLOCATE.

  Na declaração das matrizes - ALLOCATABLE

```
INTEGER, DIMENSION( : ), ALLOCATABLE :: A      ! 1D
REAL, DIMENSION( : , : ), ALLOCATABLE :: B     ! 2D
```

  Alocação de memória - ALLOCATE

```
READ*, isize
ALLOCATE(A(isize))
```

  Liberação de memória -DEALLOCATE

```
DEALLOCATE(A)
```

Expressões e operadores

- Operadores aritméticos
  - `+` (adição)
  - `-` (subtração)
  - `*` (multiplicação)
  - `/` (divisão)
  - `**` (exponenciação)

- Prioridade

Compilar e executar

gfortran <nome do programa> -o <nome do executável>

Exemplo: (nedit soma.f90)

```
PROGRAM soma          !inicia o programa chamado soma
INTEGER :: k, n, m    !declara que k,n,m são variáveis inteiras
n=1                   !atribui o valor 1 à variável n
m=3                   !atribui o valor 3 à variável m
k=m+n                 !soma n e m, e o resultado é armazenado em k
PRINT*, k             !escreve na tela a variável k
PAUSE                 !pausa a execução do programa
END                   !fim do programa soma
```

Compilar: gfortran soma.f 90 -o soma.x

Executar: soma.x

Expressões e operadores

- Operador caractere

  Utilizado para efetuar a concatenação "//", somente de variáveis caracteres.

  CHARACTER(LEN=*),PARAMETER :: string='abcdefgh'

  string(1:1) → **'a'**

  string(2:4) → **'bcd'**

  a=string(1:1)//string(2:4) → **'abcd'**

  b=string//string(2:4) → **'abcdefghbcd'**

Expressões e operadores

- Funções intrínsecas

INT(x)	converte a variável para inteiro	INT(14.8) = 14
REAL(x)	converte a variável para real	REAL(4) = 4.0
DBLE(x)	converte a variável para dupla precisão	DBLE(1) = 1.0D0
NINT(x)	converte a variável x para o inteiro mais próximo	NINT(3.8) = 4
CHAR(x)	converte a variável x inteira para caractere	CHAR(65) = A
ICHAR(x)	converte a variável x caractere para inteiro	ICHAR('A') = 65
ALOG(x)	calcula o logaritmo natural	ALOG(2.0) = 4.605170...

Expressões e operadores

- Funções intrínsecas

ABS(x)	retorna o valor absoluto de uma variável real	ABS(-14.8) = 14.8
IABS(x)	retorna o valor absoluto de uma variável inteira	IABS(-5) = 5
MOD(a,b)	mostar o resto da divisão entre a e b	MOD(5.3,2.0) = 1.3
MAX(args)	retorna o maior valor de um conjunto de variáveis inteiras	MAX(1,2,-7,4) = 4
MIN(args)	retorna o menor valor de um conjunto de variáveis inteiras	MIN(1,2,-7,4) = -7
AMAX(args)	retorna o maior valor de um conjunto de variáveis reais	AMAX(1.0,2.3,5.8) = 5.8
AMIN(x)	retorna o menor valor de um conjunto de variáveis reais	AMIN(1.0,2.3,5.8) = 1.0

Expressões e operadores

- Funções intrínsecas

SQRT(x)	retorna a raiz quadrada	SQRT(9.0) = 3.0
EXP(x)	calcula o exponencial e^x	EXP(1.0) = 2.718281746
LOG(x)	calcula o log neperiano ln(x)	LOG(2.718281746) = 1.0
LOG10(x)	calcula o log na base 10	LOG10(100.0) = 2
SIN(x)	retorna o valor do seno	SIN(1/2) = 1.0
COS(x)	retorna o valor do cosseno	COS(1/2) = 0.0
TAN(x)	retorna o valor da tangente	TAN(1/2) = 1.0
ASIN(x)	inverso da função seno	ASIN(1.0) = 0.0
ACOS(x)	inverso da função coseno	ACOS(1.0) = 1/2
ATAN(x)	inverso da função tangente	ATAN(1.0) = 1/4
LEN(x)	retorna o tamanho da variável	LEN('ABCD') = 4

Expressões e operadores

- Operadores relacionais

  == ou .EQ. (igual)
  /= ou .NE. (diferente)
  < ou .LT. (menor)
  > ou .GT. (maior)
  <= ou .LE. (menor ou igual)
  >= ou .GE. (maior ou igual)

- Operadores lógicos

  .NOT. (negação lógica)
  .AND. (conjunção)
  .OR. (disjunção)
  .EQV. (equivalência)
  .NEQV. (negação da equivalência)

Comandos para controle de fluxo

- GO TO
  - Transfere o fluxo da execução para o comando que estiver referenciado pelo rótulo n .

GO TO <n>

Exemplo:

```
      I=1
      J=I+1
      K=I+J
      GO TO 100
      k=1        ! Esse comando nunca será executado
100   PRINT*, 'K=', k
```

Rótulo → Número inteiro colocado a frente do comando e separado do mesmo por pelo menos um espaço em branco (código em formato livre) ou nas colunas 1 a 6 de uma linha (formato fixo)

Comandos para controle de fluxo

- IF ... END IF

Permite o desvio no fluxo de execução de um programa de forma condicional.

```
IF (<expressão lógica 1>) THEN
<comando 1>
...                          } comandos executados se a expr. lógica for verdade
<comando n>
ELSE IF (<expressão lógica 2>) THEN
<comando 1>
...                          } comandos executados se a expr. lógica 1 for falsa
<comando n>                    e a expr. lógica 2 for verdadeira
ELSE
<comando 1>
...                          } comandos executados se as expr lógicas 1 e 2 forem
<comando n>                    falsas
END IF
```

Comandos para controle de fluxo

- DO ... END DO
  - Permite que um bloco de comandos seja repetitivamente executado.

```
DO <variável> = <valor inicial>, <valor final>, <incremento>
<comando 1>
...
<comando n>
END DO
```

Exemplo:

```
DO i=1,10
   DO j=1,5
      PRINT*, i,j
   END DO
END DO
```

Comandos para controle de fluxo

- DO WHILE ... END DO
  - Permite que um bloco de comandos seja executado repetidas vezes enquanto a condição for verdadeira.

```
DO WHILE (<condição>)
<comando 1>
...
<comando n>
END DO
```

Exemplo:

```
i=1
DO WHILE  (i <= 10)
    PRINT*, i
    i=i+1
END DO
```

Comandos para controle de fluxo

- STOP
  - Provoca o término imediato do programa (parada incondicional)

  STOP [constante numérica] [conjunto de caracteres]

  constante numérica e conjunto de caracteres são impressos antes de encerrar-se a execução do programa
- EXIT
  - Provoca uma interrupção na execução dos comandos do bloco do DO

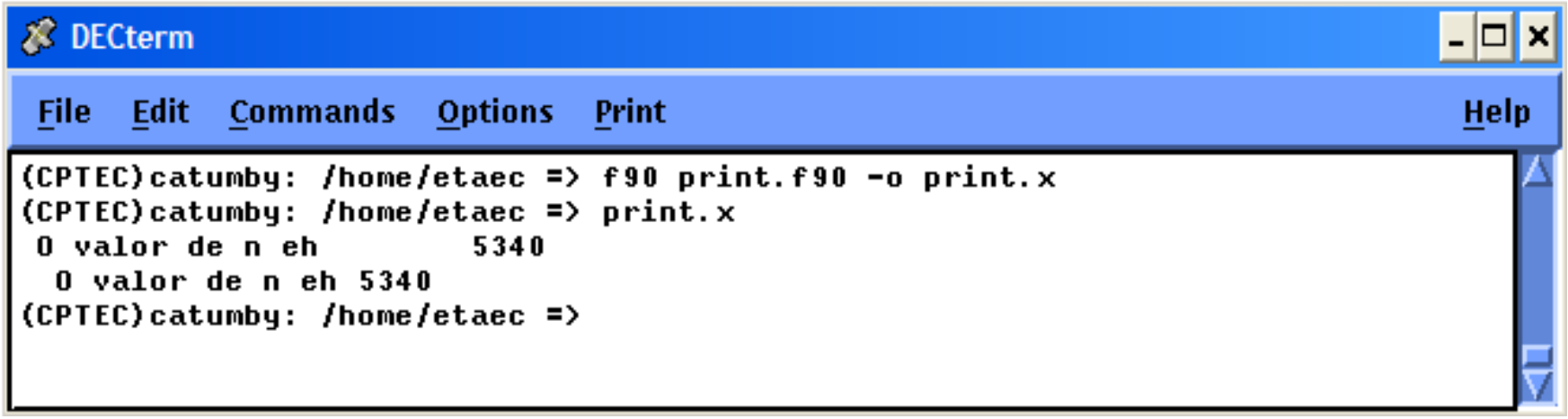
    isto é, nenhum comando existente entre EXIT e END DO é executa
- CYCLE
  - Faz com que o fluxo de execução do programa dentro do bloco do DO seja desviado para o comando END DO e uma nova iteração será executada.

Comandos e formatos de entrada e saída

- PRINT
  - Permite escrever um conjunto de caracteres no dispositivo de saída padrão (tela).

Exemplo:

```
      N=5340
      PRINT*,  'O valor de n eh ',N
      PRINT 15, 'O valor de n eh ',N
15    FORMAT (2x,I4)
```



Comandos e formatos de entrada e saí

- FORMAT
  - Permite formatar um conjunto de caracteres.

<rótulo> FORMAT (<lista de caracteres de edição>)

Caracteres de edição:

rI w	inteiros decimais
rF w.d	reais decimais, sem expoente
rE w.d	reais decimais, com expoente
rD w.d	reais decimais dupla precisão, com expoente
rG w.d	reais decimais, com ou sem expoente
rL w	lógicos
rA w	caracteres
rX	espaços em branco

r indica o número de dados que serão lidos ou escritos com aquele formato.
w representa o tamanho do campo a ser impresso
d indica a quantidade de dígitos à direita do ponto decimal dentro do campo de largura w

Comandos e formatos de entrada e saí

Exemplo:

```
      PROGRAM formata_dados
REAL    :: var1, var2
INTEGER :: var3
LOGICAL :: var4
CHARACTER (len=7) :: var5
REAL   :: var6

var1=2222.22
var2=44.44
var3=111
var4=.true.
var5='america'
var6=33333000

PRINT10,var1,var2,var3,var4,var5,var6
10    FORMAT(F7.2,x,F5.2,x,I3,x,L1,x,a7,x,e11.5)

PRINT20,var1,var2,var3,var4,var5,var6
20    FORMAT(F7.1,x,F4.2,x,I2,x,L4,x,a4,x,e8.2)

END
```

Comandos e formatos de entrada e saí

- OPEN
  - Permite que se associe um arquivo externo a uma unidade de entrada, saída ou ambos.

OPEN (unit=<nº unid>,file=<nome do arq>,status=<status>)

<nº unid> número inteiro não associado a outro arquivo
<status> new (arquivos novos), old (arquivos já existentes) ou unknow

Comandos e formatos de entrada e saí

- READ
  - Permite a leitura de um conjunto de valores de um dispositivo de entrada.

READ (<unidade>,<formato>) <var>

<unidade> número inteiro (arquivo)/ * (entrada via teclado)
<formato> número inteiro (rótulo)/ * (livre)

READ (5,*) → dispositivo de entrada padrão (teclado)

Comandos e formatos de entrada e saí

Exemplo:

```
PROGRAM le_dados_teclado

REAL :: x,y,z
REAL :: med

PRINT*, 'Entre com 3 valores...'
READ(*,*) x
READ(*,*) y
READ(*,*) z

! Calculo da media
med = (x+y+z)/3.

! Impressao do resultado no monitor
PRINT*, 'resultado: ',med

END PROGRAM le_dados_teclado
```

Comandos e formatos de entrada e saí

- WRITE
  - Permite que se escreva um conjunto de valores em um dispositivo de saída.

WRITE (<unidade>,<formato>) <var>

<unidade> número inteiro (arquivo)/ * (monitor)
<formato> número inteiro (rótulo)/ * (livre)

WRITE (6,*) → dispositivo de saída padrão (monitor)

Comandos e formatos de entrada e saída

Exemplo:

```
  PROGRAM le_dados_teclado
REAL :: x,y,z
REAL :: med

PRINT*, 'Entre com 3 valores...'
READ(*,*) x
READ(*,*) y
READ(*,*) z

! Calculo da media
MED = (x+y+z)/3.
```

Acrescentar no programa anterior

```
! Impressao do resultado no monitor
WRITE(*,*) 'resultado: ', med

! Impressao do resultado no arquivo
OPEN(unit=12,file='media.txt',status='new')
WRITE(12,10) 'resultado: ',med
10 FORMAT(a11,f10.2)

END PROGRAM le_dados_teclado
```

Comandos e formatos de entrada e saída

- WRITE/READ + DO Implícito
  - DO implícito é usado em comandos de entrada ou saída de dados.

Exemplo:

```
PROGRAM le_dados_teclado

REAL, DIMENSION(4) :: X
INTEGER :: i

! Lendo os valores de x no arquivo fort.10
READ(10,*) (X(i),i=1,4)

 ! Impressao de X no monitor
WRITE(*,*) 'X: ', (X(i),i=1,4)
WRITE(*,20) (X(i),i=1,4)
20 FORMAT(2f10.3)

END PROGRAM le_dados_teclado
```

OBS: Criar antes o arquivo txt com 4 valores (um em cada linha)

Comandos e formatos de entrada e saída

- REWIND

  – De forma semelhante a uma releitura, re-**escritura ou verificação** por leitura de um registro. O comando REWIND pode ser usado para reposicionar um arquivo, cujo número de unidade é especificado pela expressão escalar inteira <u>. Se o arquivo já estiver no seu início, nada ocorre.

    REWIND ([UNIT=]<u>)

Sub-rotinas

As sub-brotinas podem conter quaisquer tipos de comandos como imprimir resultados, abrir arquivos ou executar cálculos. As sub-rotinas podem **‘chamar’** outras sub-rotinas ou funções.

- Chamada

  CALL <nome da sub-rotina> (lista_de_parâmetros)

- Estrutura

  SUBROUTINE <nome_da_su-brotina> (lista de parâmetros)

  <definição e declaração das variáveis e constantes locais>

  <seqüência de comandos>

  RETURN

  END

Sub-rotinas

Exemplo:

```
PROGRAM subr

REAL :: x, y, result

READ*, x
READ*, y

! Chama sub-rotina
CALL soma(x,y,result)
PRINT*, x, y, result

END PROGRAM subr

SUBROUTINE soma (a,b,c)

REAL, intent(in) :: a,b
REAL, intent(out) :: c

c=a+b

END SUBROUTINE soma
```